Developing Scalable Applications using Portable Extensible Toolkit for Scientific Computation (PETSc)

Shrirang Abhyankar
abhyshr@mcs.anl.gov

Mathematics and Computer Science Division
Argonne National Laboratory

SERC@IISc-02/10/12

Outline

Outline

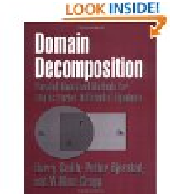
What is PETSc?

Portable Extensible Toolkit for Scientific Computation

- High performance library for the scalable (parallel) solution of scientific applications
- Developed at Argonne National Laboratory
- Mostly used by researchers in PDE applications
- Free for anyone to use, including industrial users
- Hyperlinked documentation and manual pages for all routines
- Many tutorial-style examples
- Support via e-mail `petsc-maint@mcs.anl.gov`
- Download from http://www.mcs.anl.gov/petsc

History of PETSc

- Begun in September 1991 as a platform for experimentation



- More than 60,000 downloads since 1995 (version 2)
- About 400 downloads per month

- Awards
    - Top 100 R & D award in 2009
    - Cited as the Top 10 computational science accomplishments of DOE in 2008

Portable Extensible Toolkit for Scientific computing

- Architecture
  - tightly coupled (e.g. Cray XT5, BG/P, Earth Simulator)
  - loosely coupled (network of workstations)
  - GPU clusters (many vector and sparse matrix kernels)
  - Clusters with shared memory nodes
- Operating systems (Linux, Unix, Mac, Windows)
- Any compiler
- Real/complex, single/double/quad precision, 32/64-bit int
- Usable from C, C++, Fortran 77/90, Python, and MATLAB

Portable **Extensible** Toolkit for Scientific computing

Interface for other HP libraries

- BLAS, LAPACK, BLACS, ScaLAPACK, PLAPACK
- MPICH, MPE, Open MPI
- ParMetis, Chaco, Jostle, Party, Scotch, Zoltan
- MUMPS, Spooles, SuperLU, SuperLU_Dist, UMFPack, pARMS
- PaStiX, BLOPEX, FFTW, SPRNG
- Prometheus, HYPRE, ML, SPAI
- Sundials
- HDF5, Boost

Packages can be directly downloaded and installed at configure time

`--download-<packagename>=1`

Who uses PETSc?

- Computational Scientists
  - PyLith (CIG), Underworld (Monash), Magma Dynamics (LDEO, Columbia), PFLOTRAN (DOE), SHARP/UNIC (DOE)
- Algorithm Developers (iterative methods and preconditioning)
- Package Developers
  - SLEPc, TAO, Deal.II, Libmesh, FEniCS, PETSc-FEM, MagPar, OOFEM, FreeCFD, OpenFVM

What can we handle?

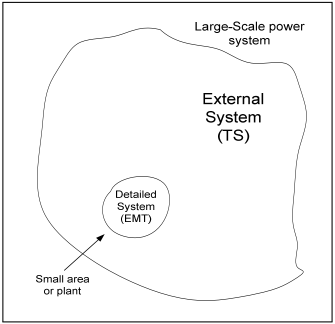
- PETSc has run problems with more than 1 billion unknowns
  - PFLOTRAN for flow in porous media
- PETSc has run on over 224, 000 cores efficiently
  - UNIC on the IBM BG/P at ANL
  - PFLOTRAN on the Cray XT5 Jaguar at ORNL

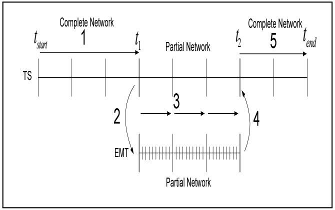
Outline

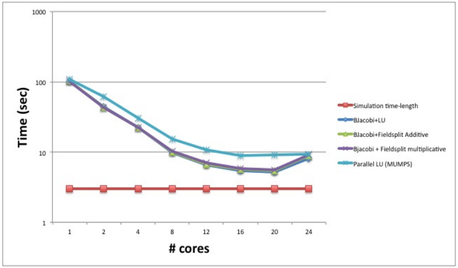
Applications of PETSc

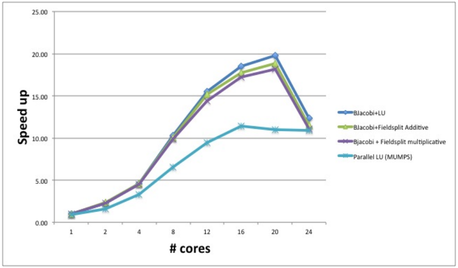
- Nano-simulations
- BiologyMedical
- Cardiology
- Imaging and Surgery
- Fusion
- Geosciences
- Environmental/Subsurface Flow
- Computational Fluid Dynamics
- Wave propagation
- Software engineering
- Algorithm analysis and design
- Electrical Power Systems
- Full list at `http://www.mcs.anl.gov/petsc/publications/index.html`

Multi-scale electrical power grid dynamics



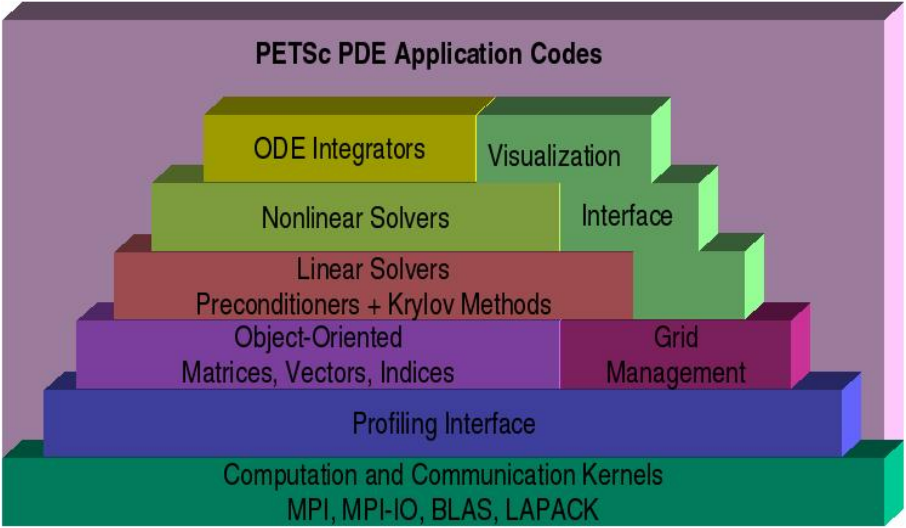






Outline

Library Organization



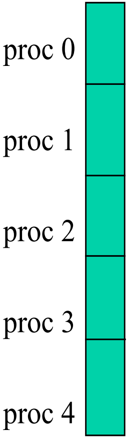
Design principles

- Linear algebra interface (Vectors, Matrices, Index sets)
- Distributed, shared nothing
  - User orchestrates communication through higher level interface
  - You almost never will have to use MPI
- Object-oriented design
  - Design based on the operations you perform
  - Example : A vector is
    - not a 1-d array but
    - an object allowing addition and scalar multiplication
- Polymorphism
  - User does not need to know the underlying implementation
- Allow solver composition to be set at run-time
  - Great for experimentation

Outline

PETSc Vectors

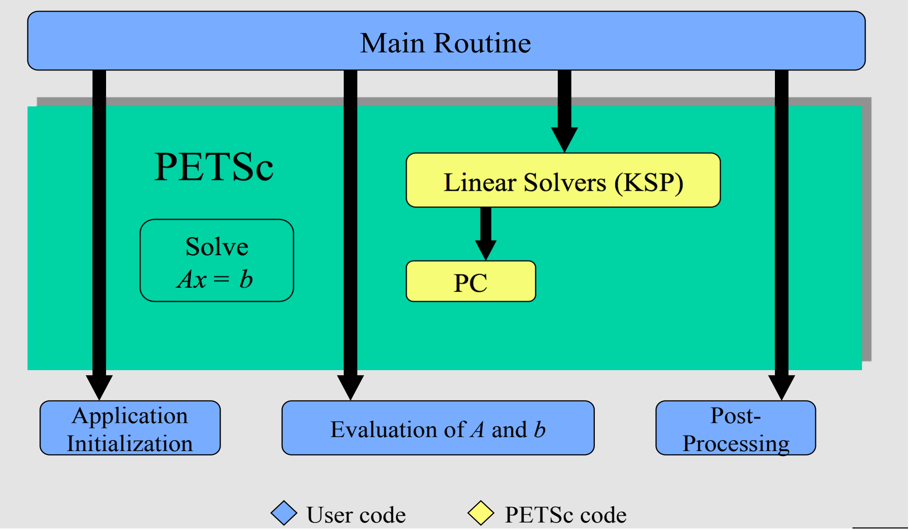
- Fundamental objects representing
  - solutions
  - right-hand sides
- Each process locally owns a subvector of contiguous global data
- Supports all vector space operations
  - `VecDot()`, `VecNorm()`, `VecScale()`
  - Unusual operations `VecSqrtAbs()`
- Entries need not be set locally
  - PETSc automatically moves data if necessary
- Supports communication of data between processes
  - `VecScatter()` – Scatter and Gather operations



PETSc Matrices

- Fundamental objects representing
  - Stiffness matrices, linear operators, Jacobians
- Each process locally owns a contiguous set of rows
- Supports many data types
  - AIJ, Block AIJ, Symmetric AIJ, Block Matrix, Dense, etc.
- Supports structures for many packages
  - MUMPS, SuperLU, UMFPack
- Polymorphism
  - Same interface irrespective of the underlying data structure.
  - User needs to only call `MatMult()`
- Can set values on any process
  - PETSc moves data to the correct process if necessary
- Supports adding custom formats

Linear solver interface



KSP

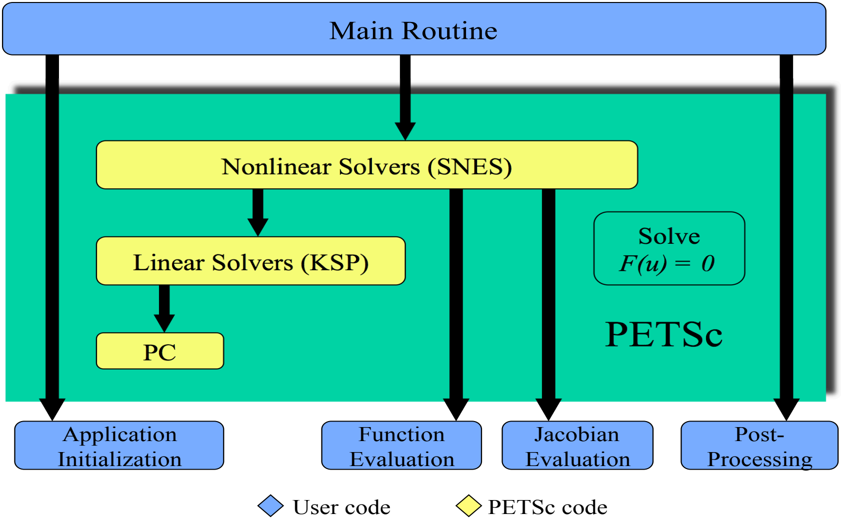
Linear solver to solve $Ax = b$

- Contains a lot of krylov subspace based solvers
  Krylov subspace: $\{b, Ab, A^2b, A^3b, \ldots\}$
  - GMRES, Conjugate Gradient, BiConjugate-Gradient, CG-squared...
- Direct solvers
  - LU, Cholesky, Incomplete LU, Cholesky (Sequential only)
- Interface for third party solvers
  - MUMPS, SuperLU, SuperLU_Dist, UMFPack
- More than 20 solvers
- Can change the solvers at run-time
  `-ksp_type <gmres, cg, cgs, bicg, preonly>`

PC

- Modify the spectrum of the linear operator to be well-behaved $(P_L^{-1} A P_R^{-1})(P_R x) = P_L^{-1} b$
- Accelerate the convergence of Krylov solvers
- More than 30 preconditioners available
- Inteface for third-party pre conditioners
- Can be changed at run-time
  - `-pc_type <lu, ilu, icc, mg, hypre>`
- Possible to combine preconditioners
  - `-pc_type composite -pc_composite_pcs ilu,jacobi`
- Supports nesting of preconditioners
  - `-pc_type <bjacobi,asm> -sub_pc_type <lu,ilu,jacobi>`
  - Can set a different preconditioned for each sub-block (a little extra effort required)
- Can add custom preconditioner `PCShell()`
- Matrix reordering schemes for reducing fill-ins
  - Minimum degree, Reverse-Cuthill, custom reordering

Scalable Nonlinear solvers (SNES)



SNES

- SNES variants
  - Line search based
  - Trust region
  - Variational inequality
- Uses underlying KSP and PC objects
- Fully customizable at run-time
  - Choose nonlinear solver, linear solver, preconditioner at run-time
- User provides
  - Code to evaluate $F(x)$
  - Optional code to evaluate Jacobian of $F(x)$
    - or use sparse finite difference approximation
    - or use automatic differentiation (ADIC/ADIFOR)

TS

- ODE integrators

$$\dot{x} = f(x, t)$$

- DAE integrators

$$\begin{aligned} \dot{x} &= f(x, y, t) \\ 0 &= g(x, y) \end{aligned}$$

- Various numerical integration schemes
    - Forward and Backward Euler, Generalized Theta, Explicit Runge-Kutta, IMEX
- Uses underlying SNES, KSP, and PC objects
- Interface for third-party package
    - Sundials

What is a DA?

`DA` is a topology interface handling parallel data layout on structured grids

- Create 1-d, 2-d, or 3-d grids `DMDACreate()`
  - Box and Star stencil types
  - Specify the degrees of freedom at each grid point.
  - Specify the number of processors along each direction.
- Provides local and global vectors
  - `DAGetGlobalVector()` and `DAGetLocalVector()`
- Handles ghost values coherence
  - `DAGetGlobalToLocal()` and `DAGetLocalToGlobal()`
- Multigrid preconditioner
  - Geometic multigrid: `-pc_type mg`
  - Algebraic multigrid: `-pc_type hypre`

Outline

Debugging

- Automatic generation of tracebacks
- Detection of memory corruption and leaks
- Optional user-defined error handlers
- Launch the debugger
  - `-start_in_debugger [gdb, dbx, noxterm]`
  - `-on_error_attach_debugger [gdb, dbx, noxterm]`
- Attach the debugger only to some parallel processes
  - `-debugger_nodes 0, 1`
- Use valgrind
  - `http://www.valgrind.org`
  - Checks memory access, cache performance, memory usage, etc.
- Check correctness of analytical Jacobian
  - `-snes_type test -snes_test_display`

Profiling

- `-log_summary`
  - Prints a report at the end of the run
  - Reports time, calls, Flops for function calls (called Events)
  - Memory usage for Objects
  - Can set stages for code profiling

Sample -log_summary

```
------------------------------------------------------------------------------------------------------------------------
Event                Count      Time (sec)     Flops/sec                          --- Global ---  --- Stage ---   Total
                   Max Ratio  Max     Ratio   Max  Ratio  Mess   Avg len Reduct  %T %F %M %L %R  %T %F %M %L %R Mflop/s
------------------------------------------------------------------------------------------------------------------------


--- Event Stage 0: Main Stage

PetscBarrier           2 1.0 1.1733e-05 1.0 0.00e+00 0.0 0.0e+00 0.0e+00 0.0e+00  0  0  0  0  0   0  0  0  0  0     0

--- Event Stage 1: SetUp

VecSet                 2 1.0 9.3448e-04 1.0 0.00e+00 0.0 0.0e+00 0.0e+00 0.0e+00  0  0  0  0  0   0  0  0  0  0     0
MatMultTranspose       1 1.0 1.8022e-03 1.0 1.85e+08 1.0 0.0e+00 0.0e+00 0.0e+00  0  0  0  0  0   0 57  0  0  0   185
MatAssemblyBegin       3 1.0 1.0057e-05 1.0 0.00e+00 0.0 0.0e+00 0.0e+00 0.0e+00  0  0  0  0  0   0  0  0  0  0     0
MatAssemblyEnd         3 1.0 2.0356e-02 1.0 0.00e+00 0.0 0.0e+00 0.0e+00 0.0e+00  0  0  0  0  0   5  0  0  0  0     0
MatFDColorCreate       2 1.0 1.5341e-01 1.0 0.00e+00 0.0 0.0e+00 0.0e+00 4.6e+01  1  0  0  0 16  36  0  0  0 74     0

--- Event Stage 2: Solve

VecDot                 2 1.0 3.2985e-03 1.0 9.56e+07 1.0 0.0e+00 0.0e+00 2.0e+00  0  0  0  0  1   0  0  0  0  2    96
VecMDot               45 1.0 9.3093e-02 1.0 1.59e+08 1.0 0.0e+00 0.0e+00 1.5e+01  0  0  0  0  5   1  1  0  0 19   159
VecNorm              112 1.0 2.0851e-01 1.0 8.47e+07 1.0 0.0e+00 0.0e+00 5.2e+01  1  1  0  0 18   2  1  0  0 64    85
```

Outline

Splitting for Multiphysics

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}$$

- Relaxation: `-pc_fieldsplit_type [additive,multiplicative,symmetric_multiplicative]`

$$\begin{bmatrix} A & \\ & D \end{bmatrix}^{-1} \qquad \begin{bmatrix} A & \\ C & D \end{bmatrix}^{-1} \qquad \begin{bmatrix} A & \\ & \mathbf{1} \end{bmatrix}^{-1} \left(\mathbf{1} - \begin{bmatrix} A & B \\ & \mathbf{1} \end{bmatrix} \begin{bmatrix} A & \\ C & D \end{bmatrix}^{-1} \right)$$

  - Gauss-Seidel inspired, works when fields are loosely coupled

- Factorization: `-pc_fieldsplit_type schur`

$$\begin{bmatrix} A & B \\ & S \end{bmatrix}^{-1} \begin{bmatrix} 1 & \\ CA^{-1} & 1 \end{bmatrix}^{-1}, \qquad S = D - CA^{-1}B$$

Python bindings and MATLAB interface

- Python bindings (petsc4py)
  - Implemented with Cython
  - Easier to write code, maintain, and extend
  - Supports all PETSc libraries
  - `http://code.google.com/p/petsc4py`
- PETSc-MATLAB interface
  - PETSc functions can be called via MATLAB code.
  - Supports almost all PETSc functionalities
  - Uses 1-based indexing (consistent with MATLAB)

Memory-efficient LU factorization

- Revise LU data structure according to the elements accessed during triangular solves
- Store L forward followed by U backwards
- Provides better Cache performance

`Typical LU data structure`

$$[L(1,:), U(1,:), L(2,:), U(2,:), ..., L(n,:), U(n,:)]$$

`Revised LU data structure`

$$[L(1,:), L(2,:), ..., L(n,:), U(n,:), ..., U(2,:), U(1,:)]$$

Support for GPU clusters

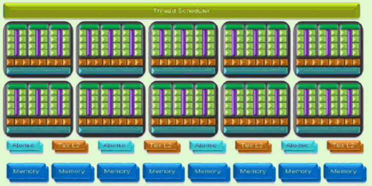


Figure: NViDia GTX 280 GPU architecture

- PETSc-3.2 (current version) supports computations on the NViDia GPUs
- Uses CUSP and Thrust libraries provided by NViDia guys
- Vec and Mat classes implemented on the GPU
- Krylov solvers come for free
- Preconditioning still an issue. Preliminary support being added for triangular solves to petsc-dev.

Clusters with shared memory nodes (Ongoing work)

- Uses POSIX threads (pthreads) for managing communication and computation within SMP node.
- Different thread pools for thread management
  - Using exclusive locks (mutex and condition variables)
  - Lockfree (atomic variables and functions)
- User control over how many and where the threads run.
- Vec and Mat class implemented using pthreads.

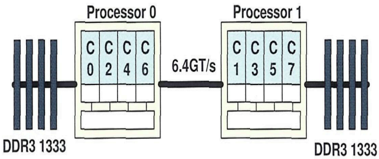


Figure: Intel Nehalem

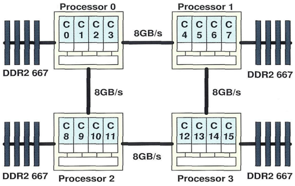


Figure: AMD Barcelona

Variational Inequalities

$$
\begin{aligned}
& f(x) = 0 \\
s.t. \qquad & xl <= x <= xu
\end{aligned}
$$

- Supports inequality and box constraints on solution variables.
- Solution methods
  - Semismooth Newton
    - reformulate problem as a non-smooth system, Newton on sub-differential
    - Newton step solves diagonally perturbed systems
  - Active set
    - solve in reduced space by eliminating constrained variables
    - or enforce constraints by Lagrange multipliers
    - sometimes slower convergence or "bouncing"

PETSc can help you

- solve algebraic and DAE problems in your application area
- rapidly develop efficient parallel code, can start from examples
- develop new solution methods and data structures
- Help installation
- debug and analyze performance
- advice on software design, solution algorithms, and performance
    - `petsc-{users,dev,maint}@mcs.anl.gov`

You can help PETSc

- report bugs and inconsistencies, or if you think there is a better way
- tell us if the documentation is inconsistent or unclear
- consider developing new algebraic methods as plugins, contribute if your idea works